## X. APPENDIX

The present claims on appeal are as follows.

1. A system of software components adapted to display an object created by an application program running under an operating system, wherein the system of software components comprises:

a first proxy component;

a second proxy component; and

a peer component for selecting either the first proxy component or the second proxy component, depending on a mode of use of the object, wherein the selection can be made during runtime, and wherein after the proxy component is selected, the selected proxy component dynamically creates a new graphics resource component for displaying the object, such that the appearance of the displayed object is substantially independent of the operating system.

2. The system as recited in claim 1, wherein the object is part of a graphical user interface associated with the application program.

3. The system as recited in claim 1, wherein the application program controls the selection of the first or second proxy components by sending an indication of the mode of use to the peer component.

4. The system as recited in claim 3, wherein the new graphics resource component is created during runtime upon selection of the first or second proxy components to replace a graphics resource component previously created for displaying the object.

5. The system as recited in claim 4, wherein the previously created graphics resource component is destroyed once replaced by the new graphics resource component.

6. The system as recited in claim 4, wherein the appearance and behavior of the object differ depending on whether the object is displayed through the selection of the first or second proxy components.

7. The system as recited in claim 1, wherein the application program is written in Java programming language.

8. The system as recited in claim 6, wherein the graphics resource components are included within a set of software components forming the Swing application program interface (API).

9. The system as recited in claim 1, wherein the operating system comprises a Windows, Unix or OS/2 computer operating system.

10. The system as recited in claim 8, wherein the first proxy component is configured for displaying the object using a Swing JTextField graphics resource component, and wherein the second proxy component is configured for displaying the object using a Swing JPasswordField graphics resource component.

11. The system as recited in claim 1, wherein the selection of the first or second proxy components depends on a status of a software flag sent to the peer component as an indication of the mode of use.

12. The system as recited in claim 11, wherein the object is adapted to respond to text entry events and wherein the status of the software flag indicates whether or not a special character is echoed when text is entered.

13. A method for displaying an object created by an application program running under an operating system, using at least one of a system of software components, which can be invoked during runtime to generate a graphical representation of the object that is substantially independent of the operating system, the method performed by the application program and comprising:

activating a first proxy component of the system of software components to dynamically generate a first graphical representation of the object during runtime;

monitoring the mode of use of the object; and

upon detecting a change in the mode of use of the object, deactivating the first proxy component and activating a second proxy component of the system of software components to dynamically generate a second graphical representation of the object during runtime, wherein the second graphical representation is distinct from the first.

14. The method as recited in claim 13, wherein the object is part of a graphical user interface associated with the application program.

15. The method as recited in claim 13, further comprising selecting the first or second software components by the application program.

16. The method as recited in claim 15, further comprising creating a new graphics resource component for generating the second graphical representation, wherein the new graphics resource component is created during runtime when the second software component is selected by the application program.

17. The method as recited in claim 16, further comprising destroying an old graphics resource component, which was previously created for generating the first graphical representation, when the first software component is no longer selected.

18. The method as recited in claim 13, further comprising varying the appearance and behavior of the object, depending on whether the object is displayed during activation of the first or the second software components.

19. The method as recited in claim 13, wherein the application program is written in Java programming language.

20. The method as recited in claim 17, wherein the graphics resource components are included within a set of software components forming the Swing application program interface (API).

21. The method as recited in claim 20, wherein the first software component is configured for displaying the object using a Swing JTextField graphics resource component, and wherein the second software component is configured for displaying the object using a Swing JPasswordField graphics resource component.

22. The method as recited in claim 13, wherein the operating system comprises a Windows, Unix or OS/2 computer operating system.

23. The method as recited in claim 13, further comprising selecting either the first or the second software component, depending on a status of a software flag associated with the object.

24. The method as recited in claim 23, further comprising the object responding to text entry events, and the software flag indicating whether or not a special character is echoed when text is entered.

25. A computer-readable storage device, comprising:

a windows-based operating system;

an application program running under the operating system;

an object created at runtime by the application program and adapted for multiple modes of use by the application program, wherein the application program is adapted for:

activating a first proxy component to dynamically generate a first graphical representation of the object during runtime that is substantially independent of the operating system;

monitoring the mode of use of the object; and

upon detecting a change in the mode of use of the object, deactivating the first component and activating a second proxy component to dynamically generate a second graphical representation of the object during runtime, wherein the second graphical representation is substantially independent of the operating system and distinct from the first graphical representation.